NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DESIGN AND IMPLEMENTATION OF THE
COMPUTER SYSTEMS DESIGN ENVIRONMENT NETWORK

by

James L. Poole

March 1985

Thesis Co-Advisor: Alan A. Ross

Thesis Co-Advisor: Norman F. Schneidewind

Approved for Public Release; Distribution Unlimited

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Design and Implementation of the Computer Systems Design Environment Network		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis March, 1985 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James L. Poole		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Navak Postgraduate School Monterey, California 93943		12. REPORT DATE March, 1985 13. NUMBER OF PAGES 92
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer aided design, CSDL, computer systems design environment, network, user manual

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis develops and implements a computer network for the Computer System Design Environment (CSDE). The CSDE network is based upon the use of the Zenith-100 (Z100) computer as an intelligent workstation. The Z100 provides direct communication with the Vax 11/780 mainframe and Prolog computer systems. Included on line in the Z100 workstation is a users manual which describes network operations. (Continued)

ABSTRACT (Continued)

The users manual is designed to take the first-time CSDE user through all phases of the CSDE process in a step-by-step manner.

Approved for public release; distribution unlimited

Design and Implementation of the Computer Systems Design Environment Network

by

James L. Poole
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1976

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
March 1985

ABSTRACT

This thesis develops and implements a computer network for the Computer System Design Environment (CSDE). The CSDE network is based upon the use of the Zenith-100 (Z100) computer as an intelligent workstation. The Z100 provides direct communication with the Vax 11/780 mainframe and Prolog computer systems. Included on line in the Z100 workstation is a users manual which describes network operations. The users manual is designed to take the first-time CSDE user through all phases of the CSDE process in a step-by-step manner.

TABLE OF CONTENTS

I. INTRODUCTION ---------------------------------- 8

A. PROBLEM STATEMENT ------------------------ 8

B. NETWORKS ---------------------------------- 8

C. CSDE AND NETWORKS ------------------------ 12

II. BACKGROUND ---------------------------------- 14

A. CSDE - INTRODUCTION --------------------- 14

1. CSDL ------------------------------------ 16

2. CSDE Overview -------------------------- 19

3. Translator ------------------------------ 21

4. Functional Mapper ---------------------- 21

5. Timing Analyzer ------------------------ 22

6. Formatter ------------------------------- 22

7. Realization Library -------------------- 23

B. COMPUTER NETWORKING --------------------- 23

III. DESIGN --------------------------------------- 30

A. CURRENT CSDE ----------------------------- 30

B. NETWORK OF CURRENT CSDE ----------------- 37

C. ASSUMPTIONS ------------------------------ 38

D. PROPOSED CSDE NETWORK ------------------- 39

IV. IMPLEMENTATION ------------------------------ 42

A. INTRODUCTION ----------------------------- 42

B. VAX - Z100 COMMUNICATION ---------------- 42

C. Z100 - PROLOG COMMUNICATION ------------- 46

D. USING THE CSDE NETWORK ------------------ 49

1. CSDE Network User Manual ------------- 49

2. CSDE Network Operation --------------- 50

3. CSDE Network Limitations ------------- 56

V. CONCLUSIONS AND RECOMMENDATIONS -------------- 59

APPENDIX A: Z100-VAX CONNECTION DIAGRAM -------- 61

APPENDIX B: CHANGING Z100-VAX BAUD RATE -------- 62

APPENDIX C: Z100-PROLOG CONNECTION DIAGRAM ----- 63

APPENDIX D: CSDE NETWORK USER MANUAL ----------- 64

APPENDIX E: USER DIRECTORY PROGRAM LISTING ----- 89

LIST OF REFERENCES ------------------------------ 90

INITIAL DISTRIBUTION LIST ----------------------- 92

ACKNOWLEDGEMENT

I would like to thank my beautiful wife Joanne for her support. She was the one who kept saying that some day this thesis would be finished.

I. INTRODUCTION

A. PROBLEM STATEMENT

The computer system design environment (CSDE) being developed at the Naval Postgraduate School involves the use of three different computers. A problem with CSDE was that none of these computers were able to directly communicate with the other. The purpose of this research project was to investigate and implement the interconnection of these computers to allow direct communication to take place. In other words, the creation of a CSDE network was the objective.

B. NETWORKS

Today, organizations are facing a computer dilemma. The problem is that organizations are not only using more computers, but are using more different types and brands. An explanation of this behavior may be broken down into three major categories: differences in computer abilities, the increased desirability of microcomputers and the lack of an organizational plan regarding the use of computers.

First, the differences in computer abilities will be considered. Computers may be classified by size and cost into four categories. Starting with the largest and most

expensive, in descending order, the categories are: mainframe, medium, mini and micro. The larger the computer the more capabilities it has and, of course, the greater the cost. Due to the various capabilities of each type, an organization may require the use of several different computers for a single application.

Second, the increased desirability of the micro will be discussed. The trend today is towards the acquisition of many small, stand-alone computer systems instead of a single mainframe [Ref. 1: p. 2]. This trend is fueled by the decreasing cost and increasing capabilities of micros. Decreased costs and increased capability result from the highly competitive field of microcomputer manufacturing. This competitiveness has produced a wide variety of microcomputer systems. A result of this trend is the increasing use of single-function computer systems and intelligent workstations [Ref. 2: p. 1]. Despite this trend, many applications exist that still require the computational power and memory capacity of the mainframe computer. For example, a CSDE program which executes in 1 minute on the Vax 11/780 mainframe computer requires 90 minutes on the Z100 microcomputer.

Third, we consider the lack of organizational planning regarding the use of computers. Organizations tend to have no overall plan regarding computer acquistions. Computers are purchased using a piece-meal approach. Each department

in the organization purchases a computer that will meet its needs at a reasonable price. No thought is given to whether each department's computer system can communicate with the others departments. This problem of computer communication exists for CSDE.

The desire for computers of all types to be able to communicate with each other has led to the creation of computer networks. Computer networks, based on the geographic distances covered, may be divided into two types: wide or local area networks. A local area network is a communications network that provides interconnection of a variety of data communicating devices within a small area. Examples of data communicating devices include: computers, terminals, peripherals, sensors and telephones [Ref. 3: p. 2]. A small area is most commonly a single building or a span of several buildings such as on a college campus. The CSDE network will fall into the local area category. Some of the typical characteristics of a local area network are: high data transmission rates (.1 to 100 Mbps), short distances (.1 to 2.5 km), and a requirement for low error rates.

Local area networks offer many benefits to justify their developmental effort. First, a network provides for the exchange of data between computers. This sharing of information precludes each computer from having to store data that it can obtain from another computer in the

network. Second, a network can allow more efficient use of mainframes. Computer applications are becoming more sophisticated and complex, requiring the capabilities of the mainframe. A network can permit efficient utilization of the mainframe by allowing the small computer systems (minis and micros) to handle the simple, local processing needs, thus freeing the mainframe to handle the large, complex processes [Ref. 4: p. 2]. Third, networks permit the sharing of expensive resources. To appreciate the need for sharing expensive resources, one must realize that although the cost of computer hardware has dropped, the cost of essential peripheral equipment, such as line printers, remains high. Fourth, networks provide for improved recovery and survivability, if they don't become too complex. The loss of a computer can be minimized in a network because of its ability to share data. Fifth, the networking of different computers prevents the user from becoming dependent upon a single computer vendor. Finally, the user needs only a single terminal to access the various computers. This benefit is particularly attractive regarding the creation of a CSDE network.

A network also has its potential pitfalls. First, interoperability may not be possible even among computers from the same manufacturer. Second, the problem of maintaining integrity, security and privacy is multiplied several times with the use of a network. Third, the

overhead incurred for interoperability may decrease both system response and performance. Finally, a loss of control in the ability to manage and enforce standards may occur as a result of networking.

The structure or topology of local area networks can be divided into two types: centralized or distributed. A centralized topology exists when one computer is the principal controller of all communication traffic. This computer controls access to the network and either acts as a switch or directs the connection of other computers wishing to communicate. Often, the central computer serves as a host for the performance of computations beyond the capabilities of other computers. In a distributed topology, on the other hand, all nodes are equal in their ability to access the network.

C. CSDE AND NETWORKS

The purpose of this research project was to create a local area network to support the CSDE system. CSDE, as will be described in the next chapter, requires the use of three separate computers. The CSDE system faced the same problem as that of many organizations. This problem, as discussed earlier, was how to get different computers to communicate in order to take advantage of each computer's capabilities.

The next chapter describes the CSDE system, its problems, the need for networking and the approach to be used to develop this network. Additionally, the goals of this research and assumptions used will be fully discussed in the next chapter.

II. BACKGROUND

A. CSDE - INTRODUCTION

The function of the Computer System Design Environment (CSDE), is illustrated in Figure 1. CSDE is a computer program taking as its input a user's description of a particular process that the user desires to automate. If the process can be automated, CSDE outputs the instructions for building the required hardware and a listing of the software needed for operation. Otherwise, CSDE outputs a listing of the errors that prevent the process from being automated.

Presently, CSDE offers the user a choice of three microprocessor families to be considered in the design of the process. The microprocessors that CSDE can examine for possible solutions are the 8080, Z80 and 8086.

CSDE is based on the work by M.N. Matelan conducted at Lawrence Livermore Laboratory in 1976 [Ref. 5]. Matelan envisioned CSDE as taking a description of a system as input and producing a microprocessor based prototype. CSDE was further developed in 1978 by LTC Alan Ross, [Ref. 6], who added the capability to build multiprocessor-based systems.

The description of the process to be automated must be expressed in the Control System Design Language (CSDL) . A description of CSDL follows. This research project assumes

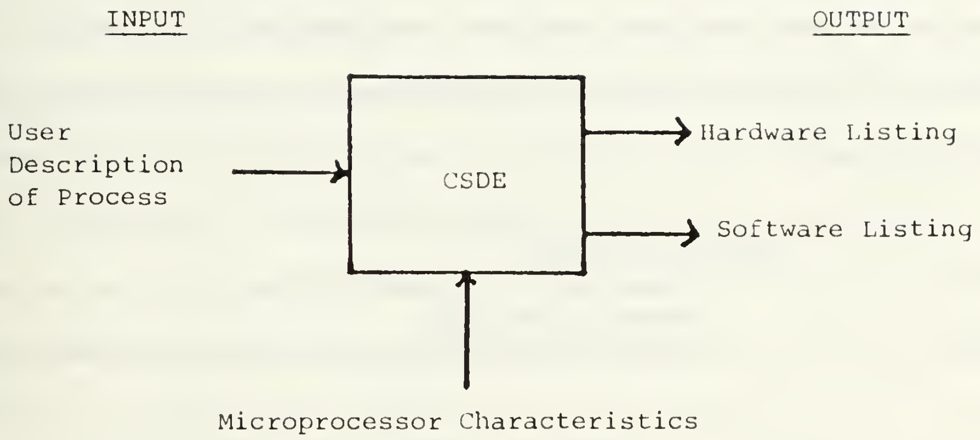


Figure 1 - Basic CSDE Operation

that the user is capable of describing his/her process in CSDL.

1. CSDL

CSDL is made up of five sections: identification, design criteria, environment, contingency and procedures.

The identification section contains a brief description of the problem under consideration, the designer's name, the version number, revision information and other comments. This section serves to identify the system but does not provide any information to CSDE. An example of an identification section is shown in Figure 2 [Ref. 7: p. 32].

```
IDENTIFICATION
     Designer: Richard Riley
     Date: 1 Nov 83
     Project: Start Controller-Version 1.4
```

Figure 2 - Example Identification Section

The Design Criteria section allows the designer to choose, to a limited degree, how the system is to be built. The designer tells CSDE to build the system based on either cost or power consumed or to select the first design that is generated. An ex-ample of this section appears in Figure 3 [Ref.8: p. 33]. The "Metric=First;" line in Figure 3 tells CSDE to select the first design generated. The "Volumes=1" line tells CSDE to examine the Z80 microprocessor as a possible solution. "Monitors=1"

indicates to CSDE that the solution's software program is to have a polled loop monitor. So far, CSDE only supports a polled monitor. Future versions will support an interrupt driven monitor, giving the user a choice between the two.

```
Design criteria Section
    Metric = FIRST ;
    Volumes = 1 ;
    Monitors = 1 ;
```

Figure 3 - Example Design Criteria Section

The Environment section contains a description of all variables in the process. These design variables are names for all input signals sensed, output signals generated and internal variables required by the process. Design variables are defined by type of signal, type of arithmetic, precision and coding. An example of this section appears in Figure 4 [Ref. 9: p. 35].

```
Environment
    Input:
        RPM,8,TTL ;Fire-Sense,1,TTL ;
        Oil Pres,8,TTL;Ignitor,1,TTL;TIT,16,TTL;
        Start-Switch,1,TTL;Reset-Switch,1,TTL;
    Output:
        SD1,1,TTL;SD2,1,TTL;SD3,1,TTL;SD4,1,TTL;
        SD5,1,TTL;SD6,1,TTL;SD7,1,TTL;SD8,1,TTL;
        Fire-Ext,1,TTL;
    Arithmetic:
        Clock,16,TTL;STAGFLG,1,TTL;
```

Figure 4 - Example Environment Section

An example of the Contingency section is shown in Figure 5 [Ref.10: p. 37]. The basic model of the process is the set of contingency-task pairs. A contingency is a condition which requires a response or some action by the processor. Each task is the action which must be carried out to respond to the contingency. The timing constraint for the contingency-task pair is also listed. It is the sum of the maximum time allowed to recog-nize that a contingency exists and the maximum time allowed to accomplish the task. From Figure 5, "When Reset-Switch(100ms) do INIT;" defines the first contingency-task pair. This line states that when the contingency, called Reset-Switch, is true the task, called INIT, must be carried out. The total time to recognize the contingency and accomplish the task is 100 milliseconds. The remaining contingency-task pairs use the variable name "EVERY" for the contingency name. "EVERY" acts as a dummy variable and is used to force the contingency-task form required in this section. The time following "EVERY" is a measure of how often the following task must be accomplished. For example, Every (100ms) do IGNAFTR; requires that the task IGNAFTER be accomplished every 100 msec.

```
Contingency:
     When Reset-Switch (100ms) do INIT ;
     Every (1000ms) do CLOCK ;
     Every (100ms) do TIT-OVR ;
     Every (100ms) do STALSTR ;
     Every (100ms) do FIRE ;
     Every (100ms) do LOWOIL ;
     Every (100ms) do OVRSPD ;
     Every (100ms) do NOIGN ;
     Every (100ms) do STAGSTR ;
     Every (100ms) do IGNAFTR ;
```

Figure 5 - Example Contingency Section

The Procedures section contains the routines which implement each contingency and task. Part of an example procedures section appears in Figure 6 [Ref. 11: p. 38].

```
Procedure
     Function Reset-Switch:
          If Reset = 1 then Reset-Switch =1 ;
               ELSE Reset-Switch = 0 ;
          End IF;
     End Reset-Switch ;
     Task TIT-OVR:
          If STARTSWITCH then ;
               Sense (TIT) ;
               If TIT>760 then SD1 = 1 FI;
               ISSUE SD1 ;
          End IF;
     End TIT OVR;
```

Figure 6 - Example Procedures Section

2. <u>CSDE Overview</u>

With the completion of the CSDL description of the problem, the designer is ready to use CSDE. Figure 7 shows a flow chart of the current implementation of CSDE. The translator takes the CSDL description and generates a

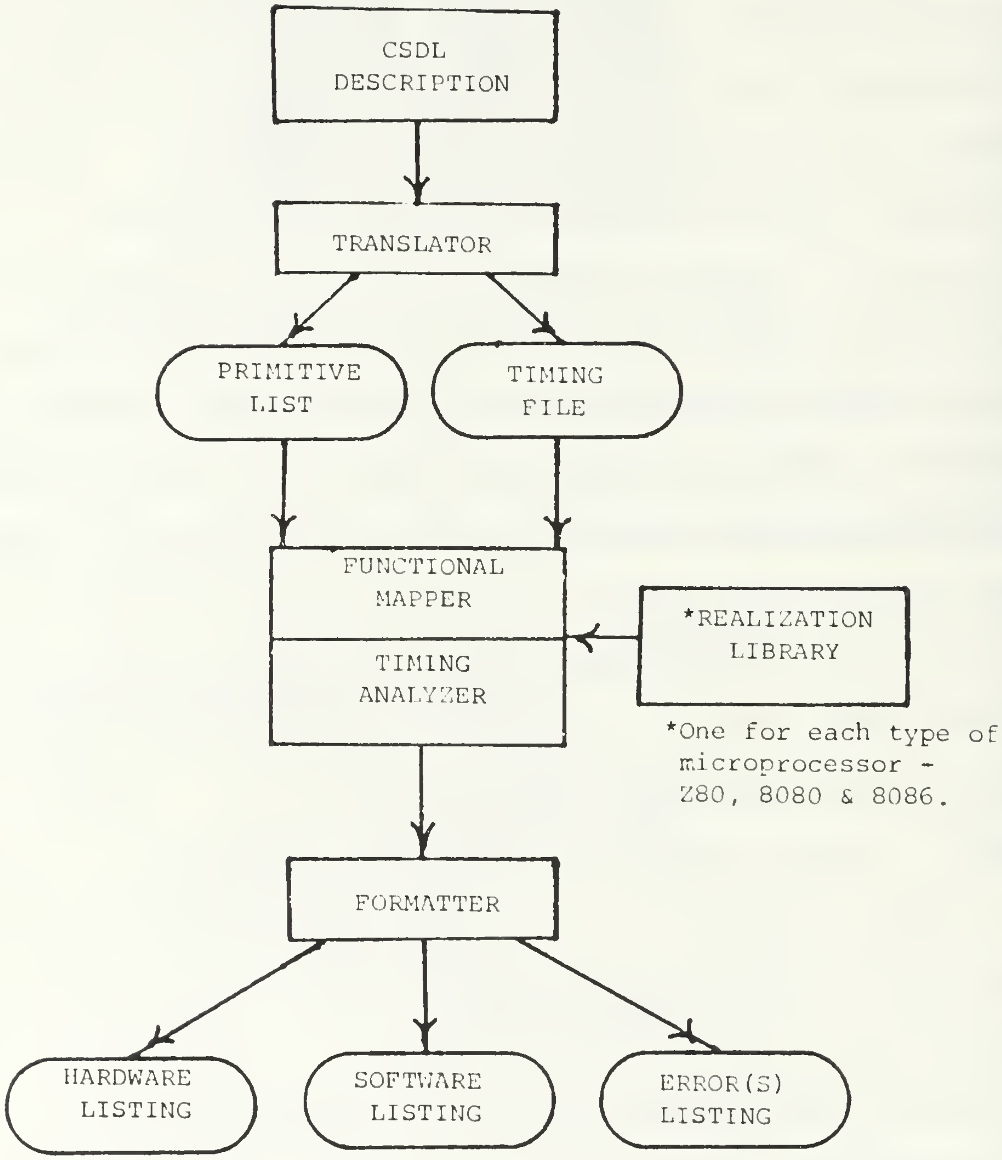


Figure 7 - Current CSDE Flowchart

primitive list and timing file. The functional mapper matches each primitive from the generated list against the realization library. The realization library contains a set of primitive realizations for the particular microprocessor being considered as a solution to the problem. The timing analyzer checks to insure the constraints from the timing file are satisfied by the timing characteristics of the matched primitives. The formatter then generates the hardware and software descriptions for the solution to the CSDL described problem.

3. Translator

The translator is a program which takes as its input the CSDL description and produces two outputs, the primitive list and the timing file. Currently, the translator is the program CSDE.EXE which is in the CSDE file. A primitive may be thought of as an instruction to the computer written in assembly language format. The primitive list results from the procedure section of CSDL. This list can be viewed as an assembly language program describing what has to be done by a computer to solve the problem. The timing file is a list of all the timing constraints that must be satisfied in order to meet the problem requirements.

4. Functional Mapper

The Functional Mapper matches each primitive from the primitive list against those within the realization library. The name of each primitive in the listing is

searched for within the realization library. After a matching name is found, the characteristics of the two primitives are compared. If any characteristic does not match exactly, CSDE will generate an error message and abort the design run.

5. Timing Analyzer

The purpose of the timing analyzer is to ensure that the set of primitive list instructions will meet the timing reauirements derived from the CSDL description by the translator. The timing analyzer also generates the instructions for the polled loop monitor. The monitor assumes that each contingency is true and all associated tasks are to be performed. The time required to perform each contingency-task pair is calculated and compared to the requirement contained within the timing file.

The monitor instructions are a set of primitives which are appended to the primitive list. The timing analyzer outputs this appended list.

6. Formatter

The outputs from the functional mapper and timing analyzer serve as input to the formatter. The formatter sequentially processes the primitive listing, using the timing analyzer output to assist in extracting the programming code for each primitive from the realization library.

The software and hardware listings are outputted to two separate files. The formatter is also responsible for dividing the primitive listing into two separate software listings if a dual processor implementation is generated.

7. Realization Library

Each realization library contains a set of primitives for a particular microprocessor. Currently, three libraries exist: one for the Intel 8080, one for the Zilog 80 and one for the Intel 8086. For CSDE to consider a microprocessor, its realization library has to be the library in use by CSDE. Each primitive within the library contains data describing its characteristics. These characteristics serve as the basis for the comparisons made by the functional mapper with regard to the primitive list.

With the discussion of the basic operation of CSDE complete, the next issue of importance involves the concepts concerned with computer networking. The next section describes a general framework for approaching the problems involved with computer networking. This general framework is then adapted to specify the model for the CSDE network.

B. COMPUTER NETWORKING

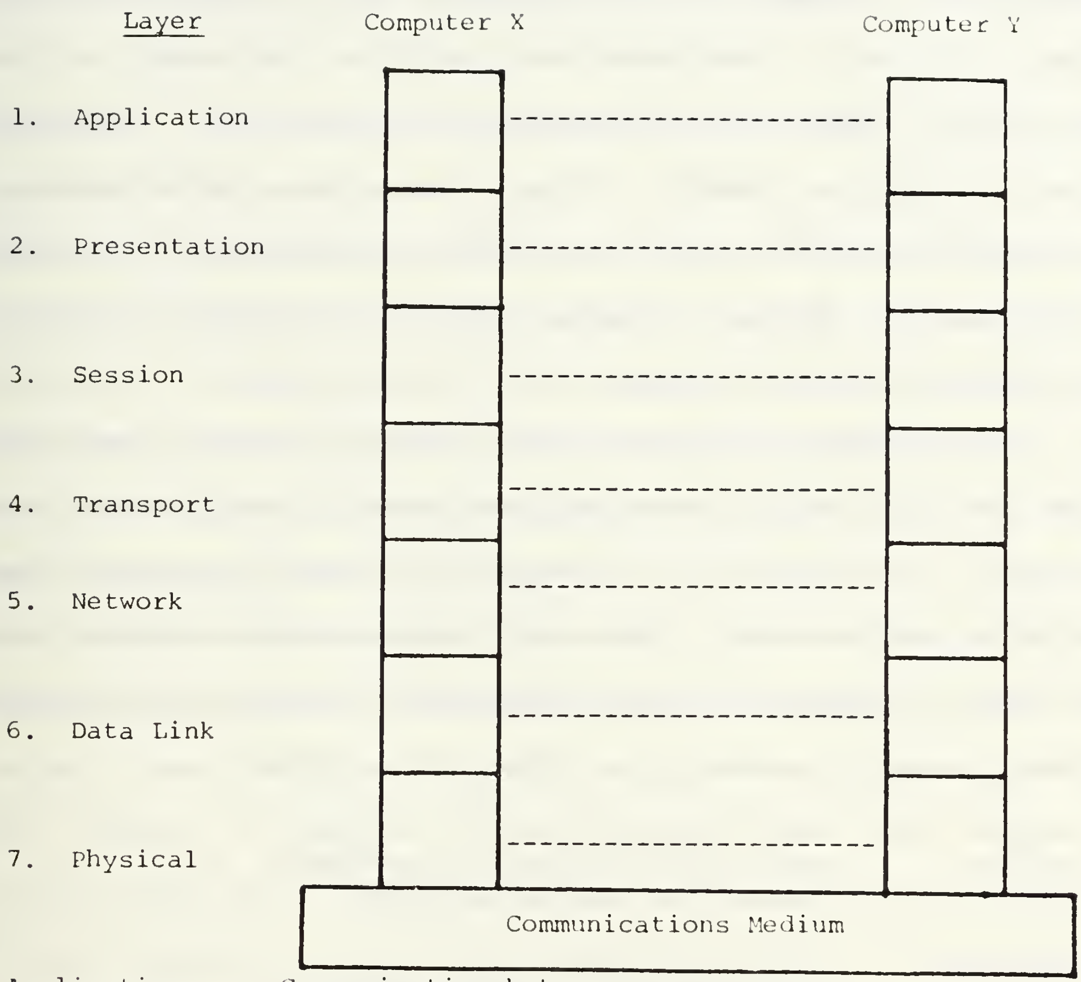
Special hardware and software are required for computers to communicate with each other. These requirements become more difficult when the communication is between heterogenous (different brands or models) computers. Some

of the problems of networking computers include differing voltage outputs/inputs, data formats and data representations.

As the use of computer networks proliferates, the-one-at-a time, special purpose solution for a particular computer network is too costly and time consuming to be practical. The only alternative is for computer vendors to adopt a common set of conventions. For this to happen, a set of international or national standards must exist [Ref. 12: p. 36].

The International Standards Organization (ISO) developed such a set of standards in 1977. This set of standards defines a network architecture called the Open Systems Interconnection (OSI) model. Its purpose is to serve as a framework for defining standards for linking different computers. The OSI model takes the communication functions and breaks them down into a hierarchical set of layers (see Figure 8). Each layer performs a set of functions required to communicate with another computer. Each depends on the layer just beneath it to perform more primitive functions, while it provides services to the next higher level.

Each layer is independent, in that a change in one will not affect any other layer. This division of function can be illustrated with the following example. The car industry is broken down into two layers: sales and manufacturing. Each group has its functions to perform. Sales depends on



Application:	Communication between user processes.
Presentation:	Format and display of user data.
Session:	Coordinates & manages the connection between two processes.
Transport:	Provides reliable end-to-end transfer of data; transparent to the user.
Network:	Routes packets through the network.
Data Link:	Provides reliable transmission of data between two points in the network.
Physical:	Transmission of bit stream over physical medium.

Figure 8 - OSI Model

Manufacturing to build and provide cars. But, Sales doesn't know how cars are built. Changes is manufacturing techniques, except for cost changes, won't affect how Sales does its job. Manufacturing won't be affected by any changes in Sale's techniques provided the same number of cars are sold as before. The lower level, Manufacturing, performs its functions and provides its services to the next higher level, Sales. Neither layer is aware of the details concerning the functions performed by the other.

For computers to communicate with one another, each must have the same set of layered functions. Communication is achieved by having the corresponding (peer) layers in each computer communicate. The peer layers communicate by means of a set of rules known as a protocol. A protocol involves syntax, semantics and timing [Ref.13: p. 37].

syntax	-	deals with areas such as data format and signal levels.
semantics	-	includes control info for coordination & error handling.
timing	-	deals with areas such as speed matching and sequencing.

Computers, no matter how different, can communicate effectively if they have the following in common:

* they implement the same set of communication functions.

* these functions exist in the same set of layers. Peer layers must provide the same functions,but not necessarily in the same way.

* peer layers must share a common protocol.

Standards are needed to assure effective communication. Standards define the functions and services to be provided by a layer (but not how it's to be done - that can differ for each computer). Standards must also define the protocols between peer layers. Each protocol must be identical for the two peer layers [Ref. 14: pp. 38-39].

The purpose of the OSI model is to take the problem of computer communication and break it down into more manageable sub-problems. For the development of the CSDE network, this research project has taken the OSI model approach and modified it.

Figure 9 illustrates the model adopted for defining the functional layers of the CSDE network. Similar to the OSI model, each layer builds upon that beneath it. Unlike the OSI model, a change in one layer may have some affect on those above it. The CSDE network consists of the following three layers:

Layer 1, the physical connection: The problems to be solved in this layer are those involved with the physical connection of the computers that will comprise the CSDE network. In addition, the question of what type of wiring and couplers to use and where on each computer to make the connection must also be addressed.

Layer 2, the communications software: This layer deals with developing the necessary software programs that will

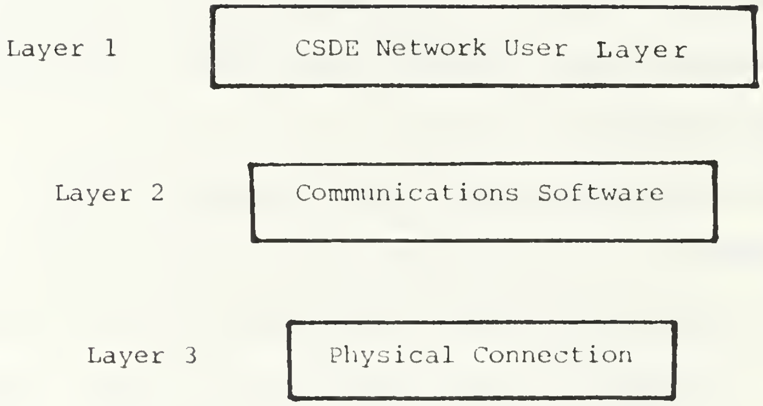


Figure 9 - CSDE Network Model

handle communications between the computers of the CSDE network. This research project will attempt to rely upon already existing communications software programs. By modifying existing software, emphasis can be placed on layer 3.

Layer 3, the CSDE user program: The development of a user friendly program that guides the user through the use of the CSDE network is the function of this layer. The goal of layer 3 is to take the results of the previous two layers and present them in a manner such that a user will view the CSDE network as a helpful, useful tool.

The next chapter describes the operation of the previous CSDE network, its problems and a proposed solution. Assumptions and rationale for the proposed design are also discussed.

III. DESIGN

A. CURRENT CSDE

A flowchart illustrating the sequence of steps required to use the CSDE system which was in effect prior to this research project is shown in Figure 10. The dotted lines represent the particular computer system on which the following steps must be performed:

STEP 1. Log onto the Vax and into the CSDE directory. This is accomplished by typing "CSDE" to the USERNAME prompt and the current password for the PASSWORD prompt.

STEP 2. Enter the Carson directory, a subdirectory of the CSDE directory.

STEP 3. Run CSDL. For this to happen, two files must be present in the Carson directory: CSDL.EXE AND DAT.DAT. The CSDL description of the problem must be labelled as DAT.DAT. Only one source file can be labelled as DAT.DAT. The CSDL program takes the DAT.DAT file as input and produces two files as output. These two files are labelled PRIMFILE.DAT and IADEFL.DAT.

STEP 4. Rename PRIMFILE.DAT to PRIMITIVE.DAT.

STEP 5. Enter the Cetel directory, another subdirectory of the CSDE directory.

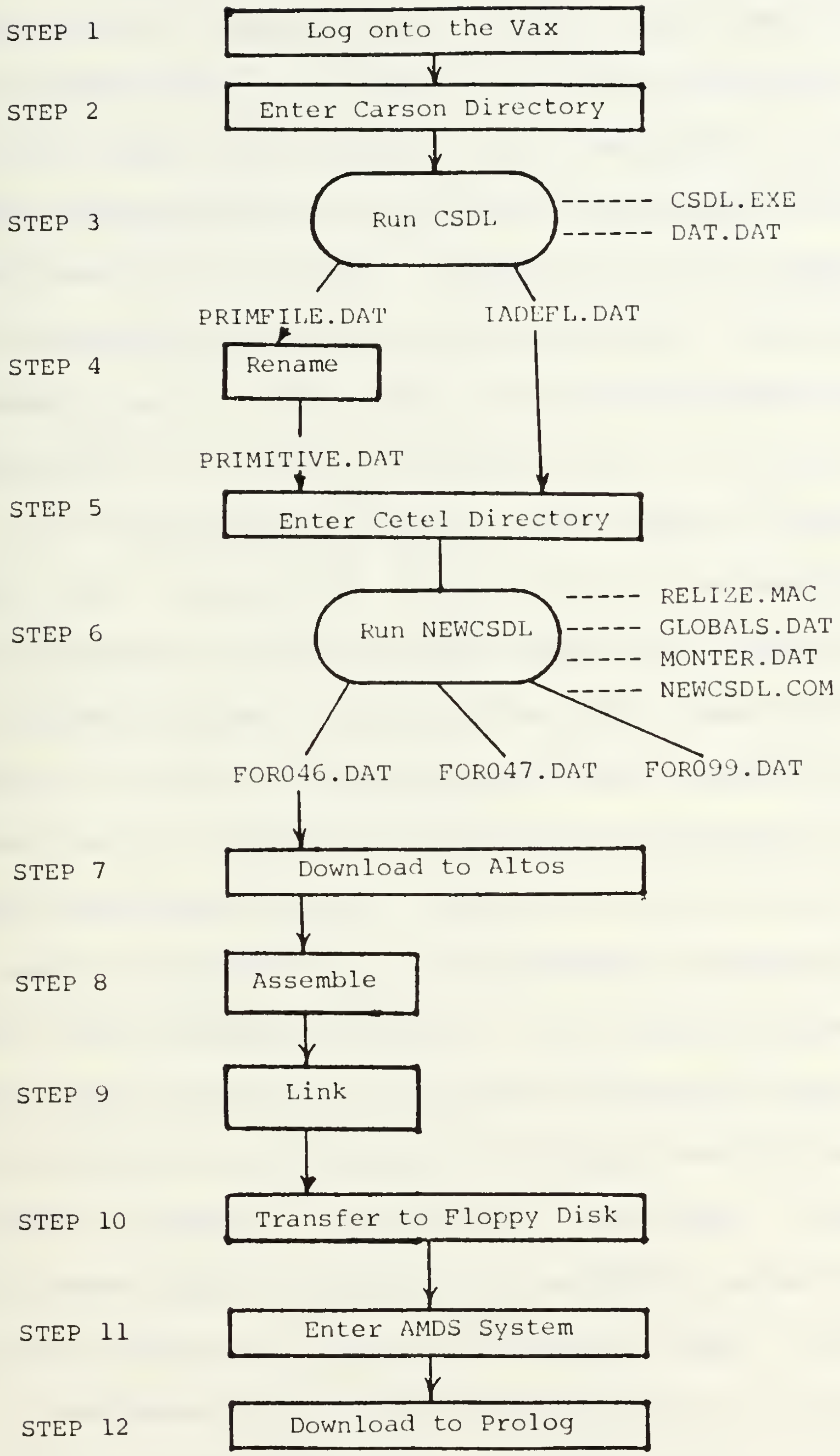


Figure 10 - Use of Previous CSDE System

STEP 6. Run NEWCSDL. Six files must be present in the Cetel directory for NEWCSDL to function: NEWCSDL.COM, GLOBALS.DAT, PRIMITIVE.DAT, IADEFL.DAT, RELIZE.MAC and MONTER.DAT. RELIZE.MAC and GLOBALS.DAT make up the realization library for the active microprocessor family and as such must be changed each time a different microprocessor is examined as a possible solution to the problem. PRIMITIVE.DAT and IADEFL.DAT (generated in step 3 above) are the intermediate form of the problem under study. These two files must be moved from the Carson directory to the Cetel directory. No other PRIMITIVE.DAT or IADEFL.DAT files from other problems can exist in the Cetel directory prior to running NEWCSDL.

NEWCSDL asks the user two questions. First, is error and status reporting to terminal desired? A yes answer drastically slows down processing. A no answer will direct all error and status information to the FOR099.DAT file, so most users should answer no. The second question is to what level of detail does the user desire to build the error report? One of three modes may be selected in creating this file. Mode 0 generates only error messages. Mode 1 generates all error messages and includes a high level trace of each module. Mode 2 generates a detailed trace including all calculations which occurred within each primitive.

A complete and correct run of NEWCSDL produces 5 files as output: FOR046.DAT, FOR047.DAT, FOR099.DAT, FOR021.DAT

and FOR063.DAT. The software listing, in assembly language for the particular microprocessor under consideration, is contained within FOR046.DAT. The hardware listing is contained within FOR047.DAT. Errors encountered during the running of NEWCSDL are listed in FOR099.DAT, as described in the previous paragraph. FOR021.DAT and FOR063.DAT are intermediate files, and are not used for any external function. They are to be erased after a correct execution of NEWCSDL.

A recommended procedure for running NEWCSDL is to use mode 0 for the initial attempt. This reduces the number of error lines listed in FOR099.DAT. If errors are present which are not immediately understood, print the error file and rerun the problem using mode 2. A printout of the mode 0 error file greatly facilitates searching the mode 2 file for more detail on each error.

Step 7. Download to Altos computer. The software listing in the FOR046.DAT file is of little use while it remains in the Vax computer. The software listing is in assembly language which requires that it be assemebled and then linked before it can be used by the particular microprocessor in question. The Vax cannot assemble or link software listings for the 8080, Z80 and 8086 microprocessors. The Altos can.

The Vax and Altos can communicate via a modem. A modem, short for modulator-demodulator, is a device used to

communicate digital data over telephone circuits. A modem must exist at both ends of the communications channel. Data from the Vax is transmitted through a modem, into the telephone circuit, to the receiving modem and finally into the receiving computer. Communications software must be used with both computers to set the parameters of the interfaces to the modems amd within the modem itself. The speed of a modem is given as a baud rate and must be set the same at both ends of the communications channel [Ref. 15: p. 54].

The procedures for downloading from the Vax to the Altos are contained in Riley's thesis, pp. 54-56. This process requires the user to make several trips between the Vax located on the fifth deck of Spanagel Hall and the Altos located on the third deck. Also, the user must obtain a copy of the software program MDM705 on an 8" floppy disk for use with the Altos.

Step 8. Assemble. Once the software listing has been moved to the Altos, it can be assembled. This is accomplished through the use of an assembler. An assembler is a software program that converts assembly language into a form that will enable the linker, step 9, to prepare the program so that it can be executed by its target computer. The Macro80 assembler will assemble code for either the 8080 or Z80.

Step 9. Link. A linker is a software program that takes

an assembled program and produces the command form of that program. The command version is the form of a program that will run on the computer.

The procedure for steps 8 & 9 depends upon the particular microprocessor being examined. Presently, research work with CSDE has been centered on the Z80 microprocessor. As a result of this work, CSDE can be run through its entirety to include building and testing using the Z80 [Ref. 16]. Therefore, this research project will give the specific procedures associated with using the Z80 as the targe microprocessor.

To assemble a file on the Altos, for example the file named TESTR.MAC, type the following command line:

M80 =TESTR/L/Z <return>

TESTR.MAC must be in Z-80 assembly language for step 8 to succeed.

The AMDS system is a network comprised of an Altos and a Prolog computer. This network enables the testing of the constructed solution proposed by CSDE. The use of the AMDS system requires that all programs start at location 4000 hex [Ref. 17: p. 22]. To link a program on the Altos so it will run on the AMDS Prolog computer, type the following command line after the assemble step (step 8):

L80 /P:4000,TESTR,TESTR/N/E/X

The program is now ready to be used in the AMDS system.

Step 10. Transfer to floppy disk. The assembled and

linked hex version of the file must be transferred to an 8" floppy disk. This step is required because the AMDS Prolog computers are located in Bullard Hall. The floppy disk must be carried over by the user in order to use the AMDS system. The file is transferred to the floppy disk by using the PIP command.

Step 11. Enter AMDS system. The AMDS system consists of a Prolog computer which is connected to a terminal and to a single user Altos. Communication is allowed between the terminal and Prolog computer for program debugging. Uploading and downloading of programs to the Prolog must take place through the Altos using the AMDS.COM program.

To enter the AMDS sytem the user performs the following three steps:

1. insert user floppy in disk drive B,
   ensure AMDS floppy in disk drive A.

2. push boot switch located on back of Altos.

3. log into disk drive B.

Step 12. Download to Prolog. First, put the Prolog in the slave mode. This is accomplished by using the attached terminal. Typing a "?" on the terminal generates a menu of available options. To select the slave mode, type "S". Second, type "AMDS" at the Altos. The AMDS program will present a menu of available options. To download a file to the Prolog, select option "G" from the menu. AMDS will ask for the filename. After the filename

is entered, AMDS will read the file into the Altos memory and download it to the Prolog. When the transfer is complete, AMDS will return to the main menu. Other options include dumping Prolog memory to the Altos screen and executing programs stored on the Prolog. A complete users manual to the AMDS system can be found in Hughes thesis, pp. 39-60.

B. NETWORK OF CURRENT CSDE

The network requirement of the previous CSDE system should be obvious from the preceeding discussion. Three computer systems that are unable to communicate with each other must be used. The use of the previous CSDE system required considerable legwork on the part of the user.

The first computer system that has to be used is the Vax located on the fifth deck of Spanagel Hall. The Vax produces the software and hardware listings.

After the software listing is produced, the user must move to the second computer system, the Altos, located on the third deck of Spanagel Hall. The required modem connections are made and the software listing is transferred from the Vax to the Altos. Once in the Altos, the listing is assembled, linked and placed on a floppy disk.

The floppy is then carried over to Bullard Hall and inserted into the third computer system, the AMDS system.

Once in AMDS, the listing can be downloaded into a Prolog computer for testing and evaluation.

The three nodes of the current CSDE system are connected by the physical movement of the user from one node to another. Each run of CSDE requires that all steps be repeated starting from the beginning. Allowing the user to perform all the steps of CSDE from one workstation would save a great deal of time and effort. A CSDE network with a single user workstation is the goal of this research project.

C. ASSUMPTIONS

This research project makes three assumptions regarding the development of the CSDE network.

First, the user knows how to describe his problem in CSDL language, including proper formatting.

Second, the user is familar with basic computer knowledge and terminology. For example, the user knows about the concept of a file, the difference between a keyboard and a screen, has some knowledge concerning microprocessors and the basic steps involved in turning assembly language into a form usable by a computer.

Finally, the user knows how to build the proposed system based on the hardware listing produced by CSDE. The Prolog library enables the user to assemble the chips and other parts called for by the hardware listing into cards which

can then be inserted into the Prolog backplane. Once all the cards have been inserted, the CSDE proposed solution can be tested.

D. PROPOSED CSDE NETWORK

This research project started with the objective of developing the Z100 computer as the intelligent workstation for the CSDE network. The Z100 is a general purpose computer capable of using either of two built in microprocessors: the 8085 or 8088. It also offers a choice of operating systems: CP/M85 or ZDOS, respectively.

Examination of the previous CSDE network resulted in the following conclusion regarding computer pairs. Of the three computers in CSDE, only two pairs require direct communications with each other: Vax-Altos and Altos-Prolog.

Use of the Z100 as the intelligent workstation combined with the two pairs of computers requiring direct communications resulted in the design of the proposed CSDE network illustrated in Figure 11. The Z100 replaces the Altos and provides communication with either the Vax or Prolog.

The 8085 microprocessor and CP/M85 operating system of the Z100 will be used for CSDE network implementation. This decision was made due to the simularities that exist with the CP/M operating system of the Altos and that of the Z100's CP/M85.

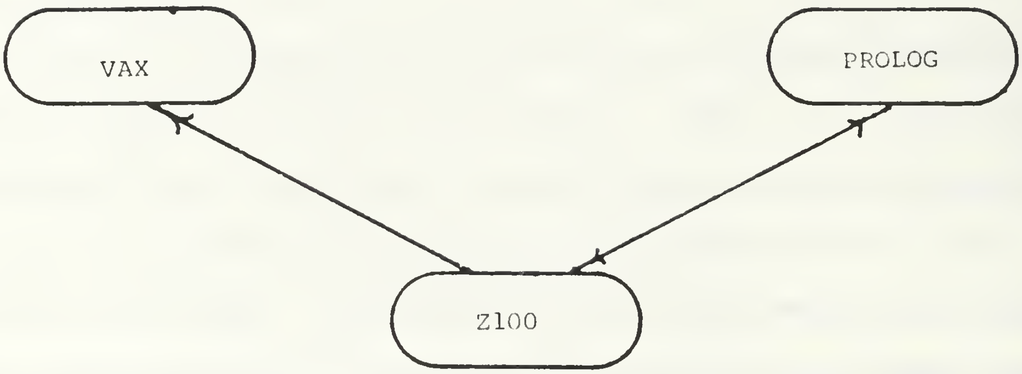


Figure 11 - CSDE Network Design

The next chapter describes the development of the CSDE network and its implementation. How the computers are connected and the operation of the CSDE network are discussed.

IV. IMPLEMENTATION

A. INTRODUCTION

This chapter deals with the implementation of the CSDE network by breaking it down into three major areas. First, the process of developing the Vax-Z100 communication is described. Second, the Z100-Prolog communication development is discussed. Third, the operation of the CSDE network, including the development of the User Manual and network limitations, is described.

B. VAX - Z100 COMMUNICATION

A direct communication line to the Vax computer was made available for this project by the Naval Postgraduate School's Department of Computer Science. Two problems had to be solved in order to achieve effective communications between the Vax and Z100 computers. The first involved making the physical connection to the Z100. The second problem was the development of a software program to provide for effective communications.

The problem of the physical connection was broken down into two categories. First, what type of coupling is required for connection of the Vax communication line to the Z100? This question was answered by the use of a standard RS-232 connector cable. Appendix A contains the detailed

information required to make this connection. The second, and more difficult, problem was having to decide where to make the connection on the Z100. The Z100 has two connections (ports) available for communications with external devices such as printers, modems or other computers [Ref.18: p. 2.82]. Such a port is called an SIO, which is short for Serial Input/Output. These ports are labelled as SIO ports A and B [Ref.19: p. 2.82].

Port B was choosen for connection to the Vax. Port A was choosen for connection to the Prolog. This pairing was made for two reasons. First, port A was designed to be used to connect to a line printer which is similar to the connection between the Altos and Prolog computers in the AMDS system. Second, port B was designed for modem use and its use eased the problem of finding a software program enabling Vax-Z100 communication.

With the physical connection solved, the problem of communications software remained. The work done by Riley and his use of the MDM705 software program for communication, via a modem, between the Vax and Altos computers directed me toward also using MDM705 for Vax and Z100 communications [Ref.20: p. 55]. Pursuing this line of research, I found that a software program providing for mainframe communication existed for the Z100. The name of this program is MDM740.

MDM740 is made up of three files, of which all must be present to create an effective communication program. M74800.ASM is the file that must be altered whenever a change in baud rate is desired. After the changes have been effected, M74800.ASM is assembled using the Z100 CP/M 85 assembler to create M74800.HEX. MDMLINK.COM, the second required file, links M74800.HEX with the third file, MDM740.COM. Appendix B describes the process for making the necessary changes for appropriate baud rate and the creation of a communication program.

I named the resultant communication program VAXLINK.COM. With the Z100 under the control of the CP/M 85 Operating System, the user only has to type the word "vaxlink" followed by a carriage return to invoke this communication program. VAXLINK then handles all aspects of Z100-Vax communications.

Three problems were encountered during the development of VAXLINK. The first problem was the lack of adequate documentation describing how to change M74800.ASM for various baud rates. This problem was solved by going through each line of the program, determining where and what changes had to be made and providing the necessary documentation. The second problem was determining the appropriate setting for SIO port B to correspond with the desired baud rate. The Z100's SIO ports are built around the MC2661 Enhanced Programmable Communication Interface

(EPCI) chip. This chip comes in three versions. Each version has a different setting for a particular baud rate [Ref.21: p. 76]. Through investigation of Z100 documentation I was able to determine that the Z100 used by this project contains version B. The baud rate characteristics of Set B of Table 1 contained within the Z100 Technical Manual Appenices are used [Ref.22: p. 76]. The four bit number obtained is converted to hex and used as illustrated by appendix B. The third problem consisted of finding a baud rate that permitted proper operation between the Vax and Z100. Through several series of trail and error, the 2400 baud rate was found to be successful. For each change in baud rate, the procedure listed in Appendix B was followed.

The choice of SIO port B for the Vax connection was fortunate because MDM740 assumes the use of this port. Therefore, no changes had to be made to the addresses of the data and status registers [Ref. 23: p. 2.82].

An important user's note concerning use of the Vax from the Z100: a typing mistake has to be corrected by the use of the delete key. Any other attempt, for example the use of the backspace key, will not be picked up by the Vax and will result in an error message.

VAXLINK permits communication between the Z100 and Vax without requiring that any special programs be on the Vax. This feature adds to the flexibility of the CSDE network in

that VAXLINK can be used on any pair of similar Z100 and Vax computers. This flexibility should greatly facilitate the transplanting of CSDE networks to other locations.

C. Z100 - PROLOG COMMUNICATION

The Z100-Prolog connection faced the same two major problems as that of the Vax-Z100: physical connection and communication software. The effort required to correct these problems was significantly aided by the work of Hughes in the development of AMDS [Ref. 24].

SIO port A of the Z100 was choosen for connection to the Prolog because of its designed use as a line printer interface. This design characteristic of port A is very similar to that of the Altos-Prolog connection of the AMDS system [Ref. 25: pp. 24-26]. Appendix C contains the detailed information required for this connection.

The AMDS system consists of a single user Altos connected with a Prolog computer. The Prolog is also connected to a separate terminal set at 9600 baud, via a spare SIO port on the Prolog. The purpose of the spare terminal is to provide the user with direct communication to the Prolog for program debugging and testing [Ref. 26: p. 54].

The software program, AMDS.COM, permits the downloading and uploading of programs to the Prolog from the Altos. AMDS.COM handles the interfacing and protocols, thereby

providing effective communication between the Altos-Prolog and Prolog-terminal pairs. A complete user's guide to the AMDS system can be found in Hughes' thesis, pages 39-60.

A limitation of AMDS.COM is that it enables implementation and testing of 8 bit microprocessors only. Still, AMDS provides an effective testing environment for two of the three current CSDE microprocessors, the 8080 and Z80.

Two major problem areas were experienced in modifying the AMDS program to handle communications between the Z100 and Prolog computers. These areas were concerned with: 1) lack of up to date program documentation and 2) the differences between the Z100's SIO ports and those of the Altos.

Hughes wrote the AMDS program in May 1981 [Ref. 27: pp. 71-147]. The program was modified in May 1982. The present AMDS program varies from the original in five ways. First, the MDSTAT subroutine contained in the original was eliminated. Second, a subroutine for handling printing under the MP/M operating system was added. This subroutine proved to be not applicable to the single user Z100 and was eliminated. Third, the subroutines MDSIN and MDSOUT had been changed significantly. Fourth, the current program has less documentation than the original. In several instances, the current documentation differs from the original even though the affected code has remained unchanged. Five, none

of these four changes are mentioned or discussed anywhere in any AMDS program documentation. Hughes thesis provided guidance, on page 28, concerning modification of the AMDS program for use on other computers. This guidance could not be used until all the problems associated with the lack of up-to-date, satisfactory documentation were identified and resolved. A lot of effort was expended re-inventing the wheel. The modifications required for CSDE network operation and the corresponding explanatons are contained within the AMDS program.

The other major problem area was in modifying the MDSIN and MDSOUT subroutines to accomidate the Z100's SIO ports. As discussed earlier, these ports are designed around the MC 2661 chip [Ref. 28: p. 2.82]. Before rewrite of the subroutines could begin, two concepts had to be understood: the characteristics of the MC 2661 EPCI and how the subroutines operate. The MC2661 EPCI characteristics are documented in the Z100 Technical Manual Appendices [Ref. 29: pp. 74-93]. A study of these characteristics found that two modifications had to be made to each subroutine. First, the Z100 SIO port checks a different bit position of the status port than does the Altos [Ref. 30: p. 81, Table 8]. Second, the Z100 SIO port does not need to be reset as does the Altos [Ref. 31: pp. 82-83]. The first modification resulted in changing the bit positions to check while the

second modification eliminated the need for code to reset the port.

Once the subroutines were modified, the SIO port addresses of the AMDS program had to be changed, as discussed on page 28 of Hughes' thesis. The status (MSTATPT) and data (MDATAPT) port addresses were changed to OE9H and OE8H respectfully in order to reflect those of the Z100 SIO port A [Ref. 32: p. 2.82].

The AMDS program, completely modified for Z100 use, was assembled and loaded on the Z100 using the ASM and Load commands of CP/M 85.

D. USING THE CSDE NETWORK

1. CSDE Network User Manual

With the physical connections and communications software completed, a functional CSDE network existed. Two problems remained to be solved before the network could be considered complete. The first problem consisted of determining the capabilities of the CSDE network. Once these capabilities were determined, procedures for their implementation had to be developed. The second problem involved describing these procedures in such a manner so that they would be easily understood and simply to follow.

From a study of the CSDE network and interactions with CSDE users, a list of 15 major CSDE processes was developed. These 15 processes, listed in Figure 12,

summarize the major options required by a CSDE user. These options are the basis of the CSDE User Manual, Appendix D. Each option consists of a series of steps. Each step describes: what actions are to be taken by the user, expected responses from the computer, how the user can get help and descriptions of possible causes for errors. These steps are designed to be simple, straight-forward and geared toward the first-time, non-computer inclined CSDE user.

An aid to assist the CSDE user is the computer program USERDIR.COM. This program resides on the Vax in the CSDE directory. Its purpose is to simplify the creation of a user directory, discussed in the next section. Once in the CSDE directory, the user creates his own user directory by typing the following: @USERDIR (LASTNAME) <return>. A sub-directory of CSDE, named (lastname), is created. This becomes the user directory for all operations involving the use of CSDE programs and files while on the Vax. USERDIR.COM modifies the LOGIN.COM file and therefore both of these files must reside in the CSDE directory for proper operation. A listing of USER.COM is contained within Appendix E.

2. CSDE Network Operation

CSDE network operation is very similiar to the procedure described in Chapter III with four major exceptions. These exceptions are:

CSDE NETWORK USER OPTIONS:

OPTION LETTER	DESCRIPTION OF OPTION
A	RETURN TO THE Z100 UNDER THE CP/M OPERATING SYSTEM.
B	DESCRIPTION AND USE OF THE CSDE NETWORK.
C	CREATING FILES USING WORDSTAR.
D	VAX LOG ON.
E	TRANSFERING A COPY OF A FILE FROM THE Z100 TO THE VAX.
F	GENERATION OF HARDWARE & SOFTWARE LISTINGS
G	TRANSFERING A COPY OF A FILE FROM THE VAX TO THE Z100.
H	VAX LOG OFF.
I	ASSEMBLING OF Z80 SOFTWARE LISTING.
J	LINKING OF ASSEMBLED Z80 SOFTWARE LISTING FOR DOWNLOADING TO THE PROLOG COMPUTER.
K	DOWNLOADING TO THE PROLOG COMPUTER.
L	USE OF THE PROLOG TERMINAL.
M	OTHER PROLOG OPERATIONS FROM THE Z100.
N	PRINTING OUT FILES.
O	COPYING FILES ON THE Z100 BETWEEN THE HARD DISK, 5" FLOPPY AND 8" FLOPPY DISK DRIVES.

Figure 12 - CSDE Processes

1. Altos replaced by the Z100.

2. NEWCSDL.EXE replaced by CLIB.EXE.

3. All CSDE programs & files reside in one sub-directory.

4. CSDE programs & files accessed via a user directory.

In addition to replacing the Altos computer, the Z100 functions as the intelligent workstation of the CSDE network. All CSDE processes can be carried out from the Z100. To handle communications with the Vax or Prolog, the Z100 has two communication programs. VAXLINK.COM handles the Z100-Vax communications and AMDS.COM handles the Z100-Prolog pair. The user no longer has to leave his seat to use CSDE.

NEWCSDL.EXE has been replaced by CLIB.EXE. CLIB is more flexible in that the program asks the user for the names of the files to be used as inputs and what the user desires to name the resulting output files. This flexibility is achieved through the use of the Filename Table. The Filename Table is a file in which the user designates the names of the input files, the names of the output files, the level of error information and whether or not information is to be sent to the screen or to files. The Filename Table also informs CLIB in which directory the various input files reside. Those files preceeded by "[CSDE.TRYIT]" are located in the CSDE sub-directory, TRYIT. All other files are located in the user directory. An example format of the Filename Table appears in Figure 13.

```
DEBUG     =    (USER LAST NAME).DBG
LEVEL     =    0
DIAG      =    1
PRIM      =    PRIMFILE.DAT
SYMBOL    =    SYMFILE.DAT
RELIZE    =    [CSDE.TRYIT]RELZ80.MAC
MONTER    =    [CSDE.TRYIT]MONTER.DAT
GLOBALS   =    [CSDE.TRYIT]GLOZ80.DAT
INTER     =    Y
APPL      =    IADEFL.DAT
SOFT      =    (USER LAST NAME).MAC
HARD      =    (USER LAST NAME).HRD
```

Figure 13 - Example Filename Table

The Filename Table is created by the user and is named, (USER LAST NAME).CSD. The "DEBUG = " line informs CLIB what to name the error(s) output listing. "SOFT = " designates the name of the software listing and "HARD = " designates the hardware listing. The "LEVEL = " line informs CLIB how much detail is desired in the error file. The user has four choices for level of detail ranging from 0 to 3. The increasing number indicates increasing detail. Most users will only need a level of zero for the first run of CLIB. The increase in level of error detail will result in much longer execution times for CLIB. The "DIAG = " line informs CLIB whether to display its information on the screen or place it into files. A value of "1" causes the information to be placed in files and kept off the screen. The other permissable value ('0') will cause all information to be sent to the console. All users should designate a value of one for the "DIAG = " line. The

"PRIM", "SYMBOL", "RELIZE", "GLOBALS" and "MONTER" lines ask for the names of the input files: PRIMFILE.DAT, SYMFILE.DAT, RELIZATION FILE, GLOBALS FILE and MONTER.DAT, respectively. As long as these files exist as designated in the Filename Table, CLIB will take them as the desired inputs. The "INTER = " line designates whether the user wants to view the Filename Table under consideration by CLIB and make changes by interaction with CLIB. A yes answer, designated by the "Y" input, will cause CLIB to display the name and contents of the Filename Table it is currently using. Also displayed is an error column. A "*" in this column indicates that CLIB couldn't find the file(s) designated by the Filename Table. A menu is presented to the user along with the necessary instructions to allow changing all or any part of the Filename Table. Once all the correct information has been entered, indicated by a lack of "*" marks in the error column, CLIB can be executed. A successful generation of the hardware and software listings is indicated by the "result=0" message. CLIB has the ability to correct minor errors found in the input files by assuming a solution based on the remainder of the data inputs. CLIB informs the user if it has encountered any errors by the " N errors in cad80" message. If any errors are encountered the user should check the DEBUG file to make sure the generated hardware and software listings are correct.

The CSDE programs, CSDL.EXE & CLIB.EXE, and associated files are currently located within the CSDE sub-directory called TRYIT.DIR. The files that reside within TRYIT.DIR are:

RELZ80.MAC	GLOZ80.DAT	MONTER.DAT
REL8086.MAC	GLO8086.DAT	CSDL.EXE
REL8080.MAC	GLO8080.DAT	CLIB.EXE

The realization files begin with "REL" and end with the designator for the particular microprocessor. The global files are labelled in a similiar manner.

The purpose of the user directory is to allow access to the programs and files of CSDE while ensuring their integrity. Once a user is logged onto the Vax under the CSDE directory, he is instructed to create his own sub-directory of CSDE. The USERDIR.COM program facilitates the creation of the user directory by taking the user's last name as input and modifying the LOGIN.COM program. The user enters his directory by typing his last name. USERDIR.COM will inform the user if the name entered has already been used. Once in his directory, the user must transfer a copy of the DAT.DAT and Filename Table files created on the Z100 into his user directory. These two files are referred to as the user's input files. While in the user directory, the CSDL.EXE program can be run with the command: " translate ". CSDL.EXE takes the DAT.DAT file from the user directory, processes it and returns the output files back to the user directory. The CLIB.EXE program is

run from the user directory with the command: " generate ". CLIB uses the files, either in the user or TRYIT directories, as designated in the Filename Table. The output files are returned back to the user directory. The programs and files of TRYIT remain unchanged. The files of each user are kept separate through the use of the user directories. Translate and generate are alias commands from the Vax LOGIN.COM procedure. Translate specifies the running of the CSDL.EXE program. Generate specifies the running of the CLIB.EXE program.

The remainder of the CSDE network operation is the same as that of the previous system. The CSDE network offers all the advantages of the previous system without the travel and manual processing requirements.

3. CSDE Network Limitations

Despite its many advantages and overall ease of operation, the CSDE network does have limitations. Three major limitations exist which are discussed in the following paragraph.

First, certain operations of the Prolog can't be implemented from the Z100. The problem lies not in the Z100 but in the AMDS.COM program. During testing of the AMDS program, certain options that are listed as available were found no longer viable even on the original AMDS system with the Altos computer. Over the years, the AMDS program has been modified resulting in the loss of the AMDS options

L and N. Option L permitted the user to locate a particular byte sequence in Prolog memory, while option N allowed the execution of the Prolog computer from the Altos. The loss of both of these AMDS options is considered minor because once the software listing has been downloaded into the Prolog, it is much more effecient for the user to run and alter the program via the Prolog terminal. The Prolog terminal is next to the Z100 and does not require the user to move in order to use it. This limitation offers the advantage that once the program has been downloaded, the Z100 is available to be used for those CSDE processes that do not require the Prolog.

A second limitation is that the CSDE network makes use of the Z100's printer port, so there is no connection for a local printer. The CSDE network can use the Vax printer. However, the Z100 is in an environment where it is shared with others who may use its printer port for other connections. Although the Z100-Prolog connection (that makes use of the printer port) may be intact, problems may arise with Z100-Prolog communications. Another user may have disconnected the Prolog from the Z100, hooked up another device and changed the configuration of the printer port, reconnected the Prolog and not have reconfigured the printer port for Prolog opertion.

If problems arise with Z100-Prolog communications perform the following check:

1. Place the Z100 under its CP/M85 operating system.

2. Invoke the CONFIGURE command.

3. Select Option P.

4. Input the following: 9600 baud, 8 bits per character, and "NONE" to the remaining three questions.

5. Select Option X.

6. Input "P" to make the changes permanent.

This procedure should be tried when an attempt to download a file to the Prolog produces a successful message but an examination of the Prolog memory indicates that no transfer occurred.

The third limitation involves the use of the Vax computer from the Z100. The only means of editing files while connected to the Vax is through the line mode. Use of the keypad mode is not possible under the current CSDE network configuration. This limitation is minimized through the use of the Wordstar editor on the Z100 and the ability to transfer copies of files between the Z100 and Vax.

A limitation that the CSDE network shares with its predecessor is that the Prolog computer under the AMDS program can only be used for 8-bit microprocessors. As discussed earlier, this limitation will not come into play until well into the future and not until CSDE has undergone greater development.

V. CONCLUSIONS AND RECOMMENDATIONS

The objective of this thesis, an operational CSDE network based on a Z100 intelligent workstation, has been accomplished. Network operation has been tested and validated by user interaction.

The flexibility of the CSDE network has also been demonstrated. The initial CSDE network Z100 workstation had to be disassembled in order to move to another campus location. This move required the use of another Z100 in the network. The connections were made and the following software programs were transferred to the new Z100 via a 5" floppy disk: VAXLINK.COM, AMDS.COM, L80.COM, M80.COM, CSDE.MAC and MLIB.MAC. The CONFIGURE command of the Z100 CP/M85 Operating System was used to initialize the Z100-Prolog SIO port. The result was a complete restoration of the CSDE network. This move successfully demonstrated the relocation capability of the CSDE network.

The CSDE network and user manual are primarily designed for work with the Z80 microprocessor. To handle the 8080 microprocessor would require the addition of the appropriate software programs to implement assembly and linking of 8080 code. These software programs would be added to the Z100 hard disk. Options I and J would be changed, as would the filename program. The 8086

microprocessor code could be assembled and linked in a similar manner as the 8080. However, the AMDS program and the Prolog computer system would have to be replaced in order to provide the testing and validation capabilities that now exist for the 8 bit microprocessors.

The Vax-Z100 communication program does not support the screen mode of the Vax editor. An area of research that remains is the modification of the communication program to permit the screen mode of the Vax editor by the Z100, or the replacement of Modem 740 with another software package that allows the use of the screen mode.

A major area of research that remains is the modification of both communication programs to provide an interactive mode. This program would prompt the user for various inputs and then automatically carry out the corresponding processes. An example would be the automatic creation of the user directory based upon the user's response to a screen prompt.

APPENDIX A

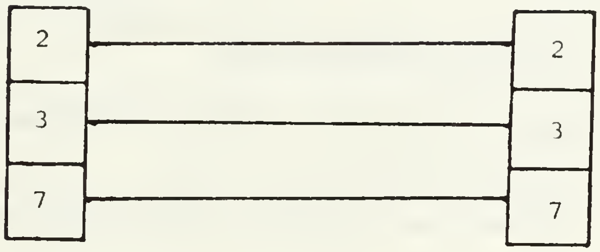
Z100-VAX CONNECTION DIAGRAM

<u>Z100</u>:
SIO Port B labelled
as J2 on back panel
of Z100.

<u>Vax</u>:
Communication line
via an RS 232C
connection.

Pin Number

Pin Number



Note: Pin connections were made by soldering a single
wire between each set of illustrated pins.

<u>Specifications</u>: 2400 Baud, 8 bits/character,
1 Stop Bit, Even Parity.

APPENDIX B

CHANGING VAX-Z100 COMMUNICATION BAUD RATE

In the event that the baud rate for Vax-Z100 communication has to be changed, the following instructions enable VAXLINK.COM to be modified to reflect the new baud rate. Baud rate may change due to an update to the MDM740.COM program, modifications to the Z100 or changes in the M74800.ASM program.

STEP 1. Use Wordstar to edit the file - M74800.ASM (Use the non-document mode).

STEP 2. Scroll down in the program to reach the following paragraph:

*** INSTRUCTIONS FOR CSDE OPERATION ***

STEP 3. Select desired baud rate and corresponding values for the two variables BAUDSPD & EPCI.

STEP 4. Enter the values for the two variables which appear immediately following the instruction block.

STEP 5. Save the edited version of M74800.ASM. Type: Control-K, wait for the Wordstar response, then type: X.

STEP 6. Assemble M74800.ASM. Type: ASM M74800.AAZ <ret>.

STEP 7. Create new Vaxlink program. Type: MDMLINK <ret>.

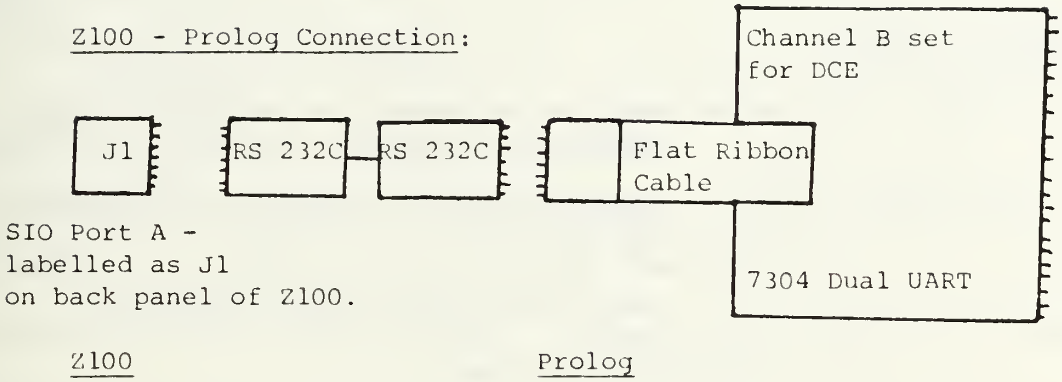
For the "input" prompt, type: MDM740.COM <ret>
For the "hex" prompt, type: M74800.HEX <ret>
For the "output" prompt, type: VAXLINK.COM <ret>

STEP 8. Have the communication line from the Vax to the Z100 changed to the new baud rate setting.

APPENDIX C

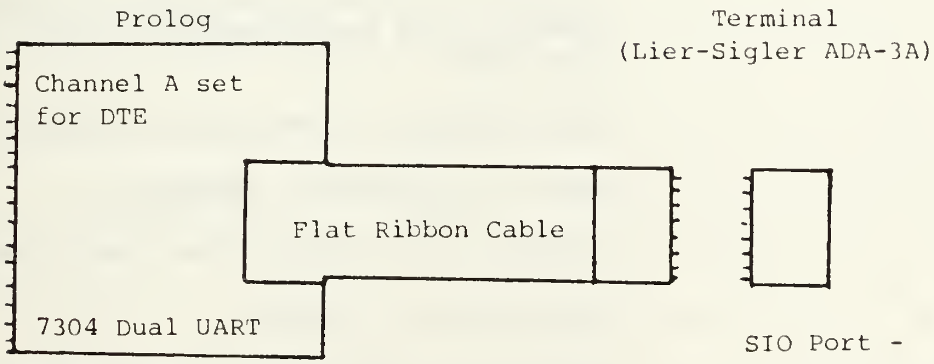
Z100-PROLOG CONNECTION DIAGRAM

Z100 - Prolog Connection:



SIO Port A Specifications: 9600 Baud, 8 Bits/Character, Handshake-None, Parity-None, Protocol-None.

Prolog - Terminal Connection:



SIO Port -
labelled as Modem
on back panel of
terminal.

Terminal Settings: 9600 Baud

Parity:	OFF	Bit 8-0:	ON
Stop:	OFF	Data:	OFF
Par:	ON	LC:	ON

APPENDIX D

CSDE NETWORK USER MANUAL

OPTION LETTER	DESCRIPTION OF OPTION
A	RETURN TO THE Z100 UNDER THE CP/M OPERATING SYSTEM.
B	DESCRIPTION AND USE OF THE CSDE NETWORK.
C	CREATING FILES USING WORDSTAR.
D	VAX LOG ON.
E	TRANSFERING A COPY OF A FILE FROM THE Z100 TO THE VAX.
F	GENERATION OF HARDWARE & SOFTWARE LISTINGS
G	TRANSFERING A COPY OF A FILE FROM THE VAX TO THE Z100.
H	VAX LOG OFF.
I	ASSEMBLING OF Z80 SOFTWARE LISTING.
J	LINKING OF ASSEMBLED Z80 SOFTWARE LISTING FOR DOWNLOADING TO THE PROLOG COMPUTER.
K	DOWNLOADING TO THE PROLOG COMPUTER.
L	USE OF THE PROLOG TERMINAL.
M	OTHER PROLOG OPERATIONS FROM THE Z100.
N	PRINTING OUT FILES.
O	COPYING FILES ON THE Z100 BETWEEN THE HARD DISK, 5" FLOPPY AND 8" FLOPPY DISK DRIVES.

OPTION A: RETURN TO THE Z100 UNDER THE CP/M OPERATING SYSTEM.

Selection of this option results in returning to normal operation of the Z100 under the control of its operating system, CP/M85.

This mode of operation is characterized by the following system prompt of CP/M85:

A>_

The Z100 may now be used in accordance with user and technical manuals.

TO RETURN TO CP/M85 WHILE ON THE Z100: Type: Control-Y.

TO RETURN TO CP/M85 FROM VAXLINK: See Option H.

TO RETURN TO CP/M85 FROM AMDS: Select AMDS option F. Type: F <return> .

OPTION B: DESCRIPTION AND USE OF THE CSDE NETWORK.

CSDE NETWORK DESCRIPTION

Figure 14 illustrates the layout of the CSDE network. Three computers make up the network: Vax, Z100 and Prolog. The Z100 serves as an intelligent workstation and provides communication with either the Vax or Prolog. Communications between the Vax and Z100 is handled by the VAXLINK program and the Z100-Prolog communications is handled by the AMDS program.

The purpose of the CSDE network is to take a problem description, generate the necessary hardware and software listing, convert the software listing for computer use and then provide a means for testing the finished solution. These four major process of the CSDE network are summarized as follows:

1. Input the problem description.

2. Generate hardware & software listings of proposed solution.

3. Convert software listing for computer use (assemble and link).

4. Testing of solution.

These four processes and the computer responsible for its implementation of each, are identified in figure 14. The CSDE network permits the user to accomplish all four processes from the Z100 workstation.

VAX: 2 PROLOG: 4

Z100: 1 & 3

Figure 14 - CSDE Network

USE OF THE CSDE NETWORK

This section explains the concepts behind the operation of CSDE and gives a step-by-step approach designed to take the first time CSDE user through the entire process.

CSDE requires two files from the user that will serve as part of the inputs, a CSDL description of the problem and a Filename Table. The CSDL description file must be named DAT.DAT. The Filename Table must have the following form:

```
Debug    =  (user last name).DBG
Level    =  0
Diag     =  1
Prim     =  PRIMFILE.DAT
Symbol   =  SYMFILE.DAT
Relize   =  [csde.tryit]RELZ80.MAC
Monter   =  [csde.tryit]MONTER.DAT
Globals  =  [csde.tryit]GLOZ80.DAT
Inter    =  y
Appl     =  IADEFL.DAT
Soft     =  (user last name).MAC
Hard     =  (user last name).HRD
```

The Filename Table informs CSDE what files to use as inputs, the location of these files and what names to give the various files produced as output.

The generation of the hardware and software listings is a two step process. Each process involves the use of one of the two CSDE programs. The first CSDE program to be used (CSDL.EXE) takes as its inputs the user's DAT.DAT file and produces four files as output. These output files are: PRIMFILE.DAT, IADEFL.DAT, SYMFILE.DAT and TRANSLATE.DAT. The user should check TRANSLATE.DAT to see if any errors were found in DAT.DAT. The second CSDE program (CLIB.EXE) takes as its inputs the results of CSDL.EXE along with three other files. These three files consist of MONTER.DAT along with the realization (REL___.MAC) and globals (GLO___.DAT) files for the particular microprocessor under consideration. The "Relize = ..." and "Globals =..." lines of the Filename Table tell CLIB.EXE what microprocessor is to be examined as a possible solution. For example, the above format of the Filename Table contains the listing for the Z80 microprocessor. Similiarly, part of the DAT.DAT file informs CSDL.EXE of the desired microprocessor.

CLIB.EXE outputs three files of interest to the user: hardware listing, software listing and a list of errors (if any). These files, as designated in the Filename Table, are as follows:

Hardware Listing .... (last name).HRD;1
Software Listing .... (last name).MAC;1
Error(s) .... (last name).DBG;1

The creation of the hardware and software listings takes place on the Vax computer within the CSDE directory. The CSDE programs and associated files are contained within a sub-directory of CSDE called TRYIT.DIR. The files that reside in TRYIT.DIR are:

RELZ80.MAC	GLOZ80.DAT	CSDL.EXE
REL8086.MAC	GLO8086.DAT	CLIB.EXE
REL8080.MAC	GLO8080.DAT	MONTER.DAT

Entering the CSDE directory upon logging on to the Vax, the user creates his own sub-directory of CSDE. This sub-directoryis called the user directory. The two user files, DAT.DAT and Filename Table, must reside within the user directory throughout the use of CSDE. The CSDE programs use the files that reside within the user directory or the TRYIT directory depending on how they were designated in the Filename Table. Those files preceeded by "[csde.tryit]" are located in the TRYIT directory. All other files are located in the user directory. All resulting output files from the CSDE programs are returned to the user directory.

Any changes to the DAT.DAT file require that the user start the process of generation of the hardware and software listings again. The outputs from the previous use of CSDL.EXE must be deleted prior to the use of the CLIB.EXE program. To eliminate confusion, the output files from the previous CLIB execution should also be deleted prior to use of CLIB with the modified DAT.DAT file.

Changes to the DAT.DAT and Filename Table files are required each time the user desires to check for a different microprocessor. The user should create a different user directory for each proposed microprocessor solution in order to avoid problems arising from the mixing of files for different microprocessors.

A user may have several different problems for CSDE consideration. A user directory for each problem may be created and contain the appropriate input files. For example, a user whose last name is Poole has three problems for CSDE. He would have to create the following user directories: POOLE1.DIR, POOLE2.DIR and POOLE3.DIR.

After the software listing is generated, a copy of it is transferred to the Z100. The Z100 will assemble and link

the software into a form suitable for the selected microprocessor. This manual assumes the Z80 is the microprocessor.

The resulting program is downloaded into the Prolog computer. The Prolog must first contain the required hardware as specified by the hardware listing.

The solution system is now complete and may be tested and patched while it is in the Prolog.

Other features of the CSDE network permit the user to make printed copies of any files and to transfer files to floppy disks for greater security and back up.

STEP-BY-STEP USE OF THE CSDE NETWORK, INCLUDING APPLICABLE CSDE OPTIONS

STEP 1. Create DAT.DAT and Filename Table files. (Option C)

STEP 2. Create User Directory. (Options E & F)

STEP 3. Generate hardware & software listings. (Option F)

STEP 4. Assemble & Link software listing. (Option G, I & J)

STEP 5. Download software to Prolog. (Option K)

STEP 6. Test complete solution. (Option L)

OPTION C: CREATING FILES USING WORDSTAR

*NOTE - 1. Any type of editor may be used. This option assumes the use of Wordstar.

2. The user is assumed to know about the use of CSDL.

STEP 1. Ensure the Z100 is under CP/M85 operation. This is indicated by the following prompt: A>_ .

STEP 2. Invoke the Wordstar program. Type: WS<return> .

Wordstar will announce itself and after a few seconds a table of instructions will appear.

STEP 3. Open a non-document file. Type: N .

Wordstar will ask you for a file name.

STEP 4. Name the file. The name may consist of 1 to 8 characters (the first must be a letter) followed by a period followed by three letters (called an extension).

The CSDE user will have to create two files: a CSDL description and a filename table. The CSDL file must be named DAT.DAT while the filename table will be named after the user's last name (up to 8 letters), followed by a period and the letters CSD. An example of a filename table for a user named Poole would be: POOLE.CSD.

STEP 5. Turn Insert off. Type: Control-V (Control-anyletter means to hold down the CTRL key and letter key together)

STEP 6. Type the file. You are now ready to type the program by following the directions in the main menu.

STEP 7. File is complete and ready to save. Type: Control-K . To save the file and return to the Z100, type the letter X. If you don't want to save the file, type the letter Q.

*NOTE - While typing the file, Control-O offers such services as changing spacing, centering of text...etc.

FOR MORE DETAILS CONCERING THE USE OF WORDSTAR, REFER TO THE WORDSTAR MICROSOFT MANUAL OF THE Z100.

THE REQUIRED FORM AND CONTENT OF THE DAT.DAT FILE IS DESCRIBED IN THE THESIS BY CARSON [Ref. 33].

OPTION D: VAX LOG ON.

*NOTE - When connected to the Vax, a typing error can only be corrected by use of the "DELETE" key.

STEP 1. Invoke the program that handles communications between the Z100 and Vax. Type: Vaxlink <return>.

Message announcing MDM740, along with copyright info and the following prompt will appear...

A>>COMMAND:_

STEP 2. Enter the terminal mode. Type: T <return> <return>.

Message announcing connection to the Vax/VMS 11/780 and prompt for Username will appear.

STEP 3. Enter the CSDE directory. For "Username:" type: CSDE <return>. You will then be asked for the password, type: (the current password) <return>.

*NOTE- If you wait too long to enter either data of Step 1, an error message will be displayed. This error message will also result from any typing mistakes left uncorrected. Start over with Step 1 by typing: <return> .

After a few seconds a welcome message and other info regarding the Vax will appear. When the "$" prompt appears you are ready to use the Vax.

FOR MORE DETAILS CONCERNING THE USE OF THE VAX, REFER TO THE VAX/VMS USER'S GUIDE, CHAPTER 1.

FOR MORE DETAILS CONCERNING THE USE OF VAXLINK, TYPE "M" AT THE COMMAND PROMPT

OPTION E: TRANSFERING A COPY OF A FILE FROM THE Z100 TO THE VAX.

STEP 1. Invoke the program that handles communications between the Z100 and Vax. Type: Vaxlink <return>.

Message announcing MDM740, along with copyright info and the following prompt will appear...

A>>COMMAND:_

STEP 2. Enter the terminal mode. Type: T <return> <return>.

Message announcing connection to the Vax/VMS 11/780 and prompt for Username will appear.

STEP 3. Log on to the Vax. See Option D concerning how to log on to the Vax.

*NOTE - Anytime you make a typing mistake while using the Vax, it can be corrected only by using the "DELETE" key!

STEP 4. Enter the directory where you want the file to reside. For most users this means to enter your user directory (see step 4 of Option F). At the "$" prompt, type: (lastname) <return>.

The "$" prompt will appear. To view the files that reside in the directory, type: dir <return> . The name of the directory and a list of the files it contains will be displayed.

STEP 5. Invoke the Vax editor. Type: EDIT/EDT <return>. The following prompt will appear...

$ File:

STEP 6. Enter the name of the file as you want it on the Vax. Select a filename that does not already exit in the directory in which you are located. Type: (filename).(ext) <return>.

Message announcing that input file does not exist along with a "*" prompt will appear. The "*" prompt means you are now in the Vax editor.

STEP 7. Enter the insert mode. Type: INSERT <return>. The cursor will move down a line and over.

STEP 8. Type: Control-T. (hold both keys down together)

You will be asked for filename to send.

STEP 9. Enter the name of the Z100 file you wish to copy to the Vax. This file must be on the Z100 hard disk (see Option O for transferring a file to the hard disk). Type: (filename).(ext) <return>.

You will be asked if you desire a time delay.

STEP 10. Type: Y.(yes, you do) The Z100 file will scroll across the screen. The transfer has been completed when the following two messages appear...

[Transfer completed]
(in Terminal-mode now)

STEP 11. Mark the end of the file. Type: Control-Z. (hold both keys down together)

An [EOB] message and "*" prompt will appear.

STEP 12. Exit from the editor. Type: EXIT <return>. The following info will be displayed and you will be returned to the Vax prompt ("$")...

[(vax directory file located in)]filename.ext;1 number of lines in file

TO SEND MORE FILES FROM THE Z100 TO THE VAX, REPEAT STEPS 4 - 12.

*HELP NOTE - The Vax panic button is the Control-C. Typing Control-C will return you to the "$" prompt which indicates the Vax command mode.

STEP 13. See Option H for logging off the Vax.

FOR MORE DETAILS CONCERNING THE USE OF THE VAX EDITOR, REFER TO THE EDT EDITOR MANUAL, CHAPTER 1. **NOTE: VAXLINK DOES NOT PERMIT THE USE OF KEYPAD EDITING.

FOR MORE DETAILS CONCERNING THE USE OF VAXLINK, TYPE "M" AT THE COMMAND PROMPT FOR THE USER'S MENU.

OPTION F: GENERATE HARDWARE AND SOFTWARE LISTINGS.

*NOTE - You must have the two input files, DAT.DAT and the Filename Table before using the CSDE programs to generate the hardware and software listings. See Option C for creating files. See Option B regarding the format of the Filename Table.

STEP 1. Invoke the program that handles communication between the Z100 and Vax. Type: Vaxlink <return>.

Message announcing MDM740, along with copyright info and the following prompt will appear...

A>>COMMAND:_

STEP 2. Enter the terminal mode. Type: T <return> <return>.

Message announcing connection to the Vax/VMS 11/780 and prompt for Username will appear.

STEP 3. Log on to the Vax. See Option D.

*NOTE - Once connected to the Vax, a typing error can only be corrected by use of the "DELETE" key.

STEP 4. If not already existing, create user directory. Type: @userdir (lastname) <return>.

STEP 5. Enter user directory. Type: (lastname) <return>.

STEP 6. Send copies of the two input files to user directory. See steps 5-12 of Option E. Name them DAT.DAT and (lastname).CSD

WITH YOUR DAT.DAT AND FILENAME TABLE FILES IN YOUR USER DIRECTORY YOU ARE READY TO USE THE CSDE PROGRAMS.

STEP 7. Run the first CSDE program. Type: translate <return>. The four files produced will reside in your user directory. To check, type: dir <return> .

STEP 8. Check TRANSLATE.DAT;1 for errors. Type: Type TRANSLATE.DAT;1 <return>. The file will scroll across the screen.

To Stop Scroll, Type: Control-S
To Resume Scroll, Type: Control-Q

If errors exist and DAT.DAT must be changed, the four files produced in step 7 must be erased by use of the "DELETE" command. Once these files have been erased, repeat step 7.

STEP 9. Run the second CSDE program. Type: generate <return>.

The CSDE file assignments will appear along with a list of 7 options (numbered: 1,2,3,4,5,6 & 9).

STEP 10. Enter your filename table file. Type: 2 <return>. You will be asked for the file-name table, type: (lastname).CSD <return> .

The CSDE file assignment will appear with your filename table inputs. All the "*" in the error column will have vanished if all of the files named in the filename table are available.

STEP 11. Execute the second CSDE program. Type: 6 <return>.

A processing volume message will appear. Allow time, depending on problem complexity, from a few seconds to a couple of minutes for this CSDE program to execute.

A successful run will be noted by: "result=0" message.

*NOTE - CLIB will attempt to correct minor errors that exist within an input file. To be sure that a successful run was accomplished, check the DEBUG file for mistakes.

STEP 12. View the resulting files. Type: Dir <return>. A list of all files that reside within your user directory will appear.

To view any of your files, see Step 8.

To print out any of your files, see Option N.

The next step of the CSDE process is to transfer a copy of the software listing down to the Z100 for assembly and linking. See Options G, I & J.

FOR MORE DETAILS CONCERNING THE GENERATION OF THE HARDWARE AND SOFTWARE LISTINGS, REFER TO CARSON'S THESIS [Ref. 34].

FOR MORE DETAILS CONCERNING THE OPERATION OF CLIB, REFER TO ROSS' THESIS [Ref. 35].

OPTION G: TRANSFERRING A COPY OF A FILE FROM THE VAX TO THE Z100.

*NOTE - While on the Vax, use the "DELETE" key to correct typing errors.

STEP 1. Invoke the program that handles the Vax and Z100 communications. Type: Vaxlink <return>.

Message announcing MDM740, along with copyright info and following prompt will appear...

A>>COMMAND:_

STEP 2. Enter the name of the file as you want it on the Z100. Type: T (filename).(ext) <return>.

If you pick a name that already exists on the Z100, then you will be asked if you desire that file to be erased. Type Y for yes, N for no. After a "no" answer repeat step 2 with a different filename.

Memory buffer available message will appear

STEP 3. Type: <return>.

Message announcing connection to the Vax/VMS 11/780 and prompt for Username will appear.

STEP 4. Logon to the Vax. See Option D.

STEP 5. Enter the directory where the Vax file to be copied to the Z100 resides. For most users this means to enter your user directory (see step 4 of Option F). At the "$" prompt, type: (lastname) <return>.

The "$" prompt will appear. To view the files that reside in the directory, type: dir <return> . The name of the directory and a list of the files it contains will be displayed.

STEP 6. Enter the name of the Vax file to be copied and sent to the Z100. Type: TYPE (filename).(ext) Control-Y.

Memory Buffer opened message will appear.

STEP 7. Send the file. Type: <return>.

File will scroll on the screen. The file has been sent when the "$" prompt appears.

STEP 8. Close the memory buffer. Type: Control-R.

Memory Buffer closed message will appear.

STEP 9. Write the file to the Z100. Type: Control-E.

Message stating that the file is still open will appear along with the following prompt...

A>>COMMAND:_

STEP 10. Close the file. Type: WRT <return>. The same prompt as in step 9 will appear.

TO SEND ANOTHER FILE, REPEAT STEPS 2 - 10.

STEP 11. Return to the Vax. Type: T <return> <return>.

STEP 12. Logoff the Vax. See Option H.

*HELP NOTES - The Vax panic button is the Control-C. Typing Control-C will return you to the "$" prompt which indicates the command mode.

The Vaxlink panic button is the Contol-E. Typing Control-E will return you to the "A>>COMMAND" prompt.

FOR MORE DETAILS CONCERNING THE USE OF VAXLINK, TYPE "M" AT THE COMMAND PROMPT FOR THE USER'S MENU.

OPTION H: VAX LOG OFF.

*NOTE - When connected to the Vax, a typing error can only be corrected by using the "DELETE" key.

*NOTE - You must log off by this procedure. The Vax does not have an automatic log off.

STEP 1. Ensure the "$" prompt exists. While on the Vax and something goes wrong or you just panic, type: Control-C. This will return you to the "$" prompt.

STEP 2. Type: Logoff <return>. Various info concerning Vax usage will appear with the last line as follows...

CSDE logged out at (date) (time).

STEP 3. Type: Control-E. The following prompt will appear...

A>>COMMAND:_

STEP 4. Type: BYE <return>. Message stating exit to CP/M will appear along with the Z100 CP/M85 prompt: A> .

You are now back to the Z100 CP/M85 operating system.

OPTION I: ASSEMBLING OF Z80 SOFTWARE LISTING.

STEP 1. Ensure the software listing has been transferred to the Z100. See Option G regarding the sending of file copies from the Vax to the Z100.

STEP 2. Invoke the Macro80 assembler. Type: M80 =(filename)/L/Z <return>.

For Step 2, filename refers to the first 1 to 8 characters before the period.

Allow the computer several seconds for the assembly. If any errors are encountered, a message will appear stating the number of errors present. The software listing must be debugged until all errors have been corrected.

FOR MORE DETAILS CONCERNING THE USE OF THE MACRO80 ASSEMBLER, REFER TO THE MACRO-80 ASSEMBLER REFERENCE MANUAL.

OPTION J: LINKING ASSEMBLED Z80 SOFTWARE LISTING FOR DOWNLOADING TO THE PROLOG COMPUTER.

*NOTE - The Z80 software listing must have been successfully assembled prior to linking. See Option E regarding assembly.

STEP 1. Invoke the linker. Type: L80 /P:4000,(FILENAME),(FILENAME)/N/E/X <return>.

Filename in Step 1 refers to the first 1 to 8 characters before the period.

A message announcing Link80 will appear. Any errors encounterd will be announced.

The abscence of any error messages signals a successful linking. The program is now ready to be downloaded into the Prolog.

FOR MORE DETAILS CONCERNING THE USE OF THE L80 LINKER, REFER TO THE MACRO-80 ASSEMBLER REFERENCE MANUAL (CHAPTER 4).

OPTION K: DOWNLOADING TO THE PROLOG COMPUTER.

STEP 1. Turn on the Prolog and its attached terminal. Prolog is turned on by a white switch on the front of its attached M281-115 Power Supply. The terminal's power switch is on the right hand side of the back panel.

Allow a few seconds for the screen to warm up. A cursor will appear when ready.

STEP 2. Slave the Prolog. On the Prolog terminal type: ? A menu of choices will appear. Type: <u>Shift-S</u>.

*NOTE - only capital letters are to be entered on the Prolog terminal.

The prolog is slaved to the Z100 - hitting any key on the prolog terminal will have no effect.

STEP 3. Invoke AMDS on the Z100. With the Z100 operating under CP/M85, type: <u>AMDS</u> <u><return></u>.

The AMDS menu will appear on the Z100 screen

STEP 4. Download to Prolog. Select option G from the AMDS menu, type: <u>G</u> <u><return></u>.

You will now be asked for the filename.

STEP 5. Designate the file to be downloaded. Enter the filename of the program that was assembled & linked in options I & J. Type: <u>(filename).hex</u> <u><return></u>.

The following message will appear:

Download Completed
MDS Start Address = ____H, Last Address = ____H

followed by the AMDS menu.

STEP 6. Return to the Z100. Type: <u>F</u> <u><return></u>.

STEP 7. Reset the Prolog. Push the reset button on the Prolog to unslave it from the Z100.

FOR MORE DETAILS CONCERNING THE USE OF THE PROLOG, REFER TO THE USER'S MANUAL CONTAINED WITHIN HUGHES' THESIS, JUNE 1981.

OPTION L: USE OF THE PROLOG TERMINAL

*NOTE - 1. The Prolog and its terminal must be turned on and allowed sufficient time to warm up. See step 1 of option K.

2. **IMPORTANT** All letters entered into to Prolog terminal must be capitalized. This is done by holding down the "Shift" key while typing the desired letter or by setting the terminal to upper case only.

STEP 1. Set up for operation. Push the Reset button on the Prolog. The following message will appear on the Prolog terminal...

```
Prolog Monitor
For Help Menu Type - ?
**
```

STEP 2. Invoke the help menu. Type: _?_ (using the shift key)

The monitor help menu will appear.

"A" - EXAMINE AND/OR CHANGE ADDRESS: This allows the user to examine and modify the memory contents of the Prolog.

STEP 3. Enter desired address. Type: _A (4 digit hex address)_.

The memory address and its contents (2 digit hex) will be displayed.

STEP 4. To change memory contents - type: _(2 digit hex)_.

To view next address & its contents - Push the "Line Feed" key.

To view previous address & its contents, type: _-_ .

STEP 5. To stop the examine option, type: _<return>_.

"S" - SLAVE TO ALTOS: The Z100 has taken the place of the Altos. This command slaves the Prolog to the Z100. Once slaved, the Prolog terminal becomes locked until the Prolog is reset.

STEP 3. Slave the Prolog to the Z100. Type: S.

STEP 4. Unslave the Prolog. Push the reset button on the Prolog.

"G" - GO COMMAND: This allows the user to execute the program stored within Prolog's memory.

STEP 3. Execute the Prolog program. Type: G 4000. This will cause execution of Prolog program which is stored at starting memory location 4000H.

STEP 4. Reset the Prolog. Push the Reset button on the Prolog.

This action resets the Prolog without changing its memory as existed just prior to pushing the reset. Remember -the program executed may have changed the initial memory that existed prior to program execution.

USE OF THE "F", "O" and "J" COMMANDS OF THE MONITOR HELP MENU: Follow the format as indicated on the help menu. If you are asked the question - "OK?" and your answer is yes, then type: Y . Otherwise, type: <return> .

FOR MORE DETAILS CONCERNING THE USE OF THE PROLOG, REFER TO THE AMDS USER'S MANUAL CONTAINED WITHIN HUGHES' THESIS, JUNE 1981.

OPTION M: OTHER PROLOG OPERATIONS FROM THE Z100

*NOTE - The Prolog must be slaved to the Z100 before using any of the options below. See step 2 of option K on how to slave the Prolog to the Z100.

STEP 1. Invoke AMDS on the Z100. See step 3 of option K.

The AMDS menu will appear on the Z100 screen.

UPLOAD PROLOG MEMORY TO HEX FILE ON THE Z100: This option permits the user to upload the Prolog memory to the Z100 and save it as a file. The file must be of the hex type.

STEP 2. Select AMDS option H. Type: H <return>. The following message will appear:

Upload (Save) MDS Memory to Disk Hex File
Filename >_

STEP 3. Enter desired filename. Type: (filename).hex <return>.

AMDS will ask for the starting and ending hex addresses by displaying the ">" prompt.

STEP 4. Enter the starting and ending hex addresses. Type: (start-4 digit hex),(end-4digit hex) <return>.

Message stating upload complete and AMDS menu will appear on the Z100 screen.

STEP 5. Return to the Z100 CP/M85 operating system. Type: F <return>.

The hex file now resides on the Z100's hard disk.

EXAMINE/SET MDS MEMORY LOCATION(S): This option allows the user to examine and modify (set) the contents of the Prolog's memory.

STEP 2. Select AMDS option I. Type: I <return>. The following message will appear:

Examine/Set MDS memory
>_

STEP 3. Enter start address. Enter the memory address to be examined by typing: (4 digit hex) <return>.

The address and data (2 digit hex) stored at that address will be displayed.

STEP 4. To change contents of displayed location, type: (2 digit hex) <return>. The 2 digit hex number is the data to be stored at that memory location.

The next memory address and its contents will be displayed.

To view the next address and its contents without changing that of the displayed address, type: <return> .

STEP 5. To stop the examine/set option, type a period: . <return>. This will return you to the AMDS menu.

STEP 6. Return to the Z100 CP/M85 operating system. Type: F <return>.

DUMP MDS MEMORY LOCATION(S) TO CONSOLE: This option is similiar to the examine/set option above except, in this case the user can view a block of Prolog memory.

STEP 2. Select AMDS option M. Type: M <return>. The following message will appear ...

```
Dump MDS Memory
>_
```

STEP 3. Enter the starting and ending addresses (up to 256 locations). Type: (start- 4 digit hex),(end-4 digit hex) <return>.

Starting address followed by the contents of the requested Prolog memory will be displayed along with the ">" prompt.

STEP 4. Stop the memory dump. Type: . <return>. This will return you to the AMDS menu.

STEP 5. Return to the Z100 CP/M85 operating system. Type: F <return>.

FOR MORE DETAILS CONCERNING THE USE OF THE PROLOG, REFER TO THE USER'S MANUAL CONTAINED WITHIN HUGHES' THESIS, JUNE 1981.

OPTION N: PRINTING OUT FILES.

*NOTE - To print out a file using the CSDE Network requires that the file reside within the user's directory on the Vax.

To transfer a file from the Z100 to the Vax, see Option E.

STEP 1. Log on to the Vax. See Option D.

STEP 2. Enter your directory. See Option F.

STEP 3. Print out the file. At the "$" prompt, type: Print (filename).(ext);(version number) <return>.

Message will appear informing you that your file is in line to be printed.

STEP 4. Pick up the print out. The print out can be picked up in Room 511 of Spanegal Hall.

OPTION O: COPYING FILES ON THE Z100 BETWEEN THE HARD DISK, 5" FLOPPY AND 8" FLOPPY DISK DRIVES.

*NOTE - The 8" floppy drive may or may not be hooked up. If it is, then it must be turned on prior to turning on the Z100.

The Z100 disk drives have the following letter designations under CP/M85:

Hard disk......."A"
5" Floppy......."C"
8" Floppy......."E"

*NOTE - Prior to using either floppy disk drive, the floppy must be formatted. Instructions for formatting are found in the Z100 CP/M85 Software Documentation Manual, pg. 2-124.

STEP 1. Ensure the Z100 is under the CP/M85 operating system.

STEP 2. Invoke the PIP command. Type: <u>PIP <return></u>.

The "*" prompt will appear, indicating PIP is ready for transfer of the file copy.

STEP 3. Enter the desired copy transfer. The form of this command is as follows:

<u>(letter of destination drive):(destination file name)= (letter of source drive):(source file name) <return></u>.

For example, to send the file POOLE.CSD from the hard disk to the 5" floppy drive, you would type the following command to the A> prompt:

C:POOLE.CSD=A:POOLE.CSD <return>

STEP 4. Exit from PIP. Once the transfer is completed, the "*" prompt will appear indicating that PIP is ready to effect another transfer. To return to the Z100 CP/M85 operating system type: <u><return></u>.

More details concernig the PIP command are found in the Z100 CP/M85 Software Documentation Manual, pg. 2-148.

APPENDIX E

USER DIREECTORY PROGRAM LISTING

```
$ON WARNING THEN GOTO END
$CREATE/DIR [.'P1']
$OPEN/WRITE OUTFILE TEMP.TXT
$WRITE OUTFILE "$"'P1'" :== SET DEF [CSDE."'P1'"]"
$CLOSE OUTFILE
$EDT/COMMAND = TEMP.EDT LOGIN.COM
$DELETE TEMP.TXT;*
$@LOGIN
$END:
```

LIST OF REFERENCES

1. Mink, A., NBS Special Publication 500-69: An Analytic Study of a Shared Device Among Independent Computing Systems, U.S. Department of Commerce, 1980.

2. Stallings, W., Local Networks: An Introduction, Macmillian Publishing Company, 1984.

3. Ibid.

4. Mink, A.

5. Matelan, M. N., The Automatic Design of Real Time Control Systems, Lawrence Livermore Laboratory, December 10, 1976.

6. Ross, A. A., Computer Aided Design of Microprocessor-Based Controllers, Ph. D. Thesis, University of California, Davis, 1978.

7. Riley, R. P., Control System Design Language Implementation of a Gas Turbine Starting Controller, M. S. Thesis, Naval Postgraduate School, Monterey, California, June 1984.

8. Ibid.

9. Ibid.

10. Ibid.

11. Ibid.

12. Stallings, W.

13. Ibid.

14. Ibid.

15. Riley, R. P.

16. Ibid.

17. Hughes, S. M., A Microprocessor Development System for the Altos Series Microcomputers, M. S. Thesis, Naval Postgraduate School, Monterey, California, June 1981.

18. Zenith Data Systems Corporation, Z100 Technical Manual Vol. 1, 1983.

19. Ibid.

20. Riley, R. P.

21. Zenith Data Systems Corporation, Z100 Technical Manual Appendices, 1983.

22. Ibid.

23. Zenith Data Systems Corporation, Z100 Technical Manual Vol. 1, 1983.

24, Hughes, S. M.

25. Ibid.

26. Ibid.

27. Ibid.

28. Zenith Data Systems Corporation, Z100 Technical Manual Vol. 1, 1983.

29. Zenith Data Systems Corporation, Z100 Technical Manual Appendices, 1983.

30. Ibid.

31. Ibid.

32. Zenith Data Systems Corporation, Z100 Technical Manual Vol. 1, 1983.

33. Carson, T. H., An Input Translator for a Computer-Aided Design System, M. S. Thesis, Naval Postgraduate School, Monterey, California, 1984.

34. Ibid.

35. Ross, A. A.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	LTC Alan Ross, Code 52RS Naval Postgraduate School Monterey, California 93943	4
4.	Prof. Herschel H. Loomis, Code 62LM Naval Postgraduate School Monterey, California 93943	1
5.	LCDR Jim Poole 2493 Stribling Lane Norfolk, Virginia 23518	1